John von Neumann Institute for Computing

Parallel I/O Optimization for an Air Pollution Model

D.E. Singh, F. García, J. Carretero

published in

Parallel Computing:
Current & Future Issues of High-End Computing,
Proceedings of the International Conference ParCo 2005,
G.R. Joubert, W.E. Nagel, F.J. Peters, O. Plata, P. Tirado, E. Zapata (Editors),
John von Neumann Institute for Computing, Jülich,
NIC Series, Vol. **33**, ISBN 3-00-017352-8, pp. 523-530, 2006.

http://www.fz-juelich.de/nic-series/volume33

Parallel I/O optimization for an air pollution model

David E. Singh[a], Félix García[a], Jesús Carretero[a]

[a]Departamento de Informática, Universidad Carlos III de Madrid, 28911 Leganés, Spain

This work presents and evaluates different I/O parallelization approaches for the Sulphur Transport Eurelian Model 2 (STEM-II), a large-scale pollution modelling application that is used to simulate air quality conditions. STEM-II is a computationally intensive application that requires of a multiprocessor environment for performing simulations in a reasonable response time. Due to the large amount of data that uses, the I/O becomes a critical factor for the application performance. This paper is focused in the study and optimization of this stage for distributed memory systems. Several parallelization approaches are presented and evaluated for a Cluster of PCs. Experimental results show that the efficient parallelization of the I/O achieves a significant reduction in the overall execution time.

1. Introduction

Nowadays, the air pollution related to high populated or industrial areas is a topic of increasingly social interest. In particular, it is especially useful the use of simulation tools for providing feedback mechanisms that allow limiting the pollutant levels. STEM-II [1] is an air quality model that simulates transport, chemical transformations, emission and deposition processes in an integrated framework. This model was successfully used for the control of the emissions of pollutants produced by the Endesa power plant of As Pontes (Spain). In addition, the STEM-II was chosen as case of study in the European CrossGrid project, proving its relevance for the scientific community for its industrial interest as well as its suitability for the high performance computing.

In terms of application performance, STEM-II is a computationally intensive application that requires a multiprocessor environment for performing simulations in a reasonable response time. In [2] several parallelization approaches were presented proving that STEM-II can be efficiently executed on a multiprocessor environment. However, the parallelization of the I/O stage was not performed, being an important bottleneck for the application performance. This topic is studied in detail in this work and efficient parallel I/O techniques are also presented. Additionally, we present two main contributions: First, an improved parallel implementation of STEM-II is shown, where the I/O output stage is completely parallelized. Second, a comparative study of different I/O parallel strategies is performed, evaluating the impact in the application performance of several parameters such as the locality degree, problem size or number of processors. This study allows introducing feedback mechanism than can be of interest to develop optimized I/O techniques.

In [3,4] the I/O requirements of several parallel applications are analyzed, showing that access patterns to non-contiguous small volumes of data are frequent. FLASH code [5] is broadly used as I/O benchmark given that it represents a real application where the access pattern is non-contiguous both in memory and in file. A similar I/O behaviour is also exhibited by the parallel implementation of STEM-II studied in this paper.

The paper organization is as follows: Section 2 describes the air quality model STEM-II, presenting its internal structure as well as its parallel implementation. The parallelization of the I/O stage is studied in Section 3. Specifically, several I/O parallel techniques are presented and discussed. The performance is analyzed in Section 4 and Section 5 summarizes the main conclusions of this work.

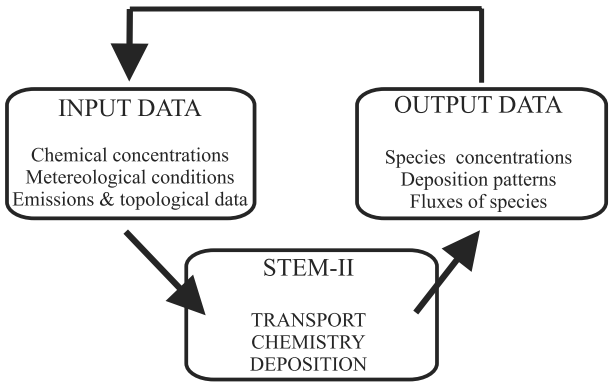

Figure 1. General block diagram of STEM-II.

2. STEM-II air quality model

Endesa power produces a power of 1400 MW obtained from the combustion of coal. This coal is a mixture made up of local lignite (with a high content in sulphur) and other foreign coals (with low sulphur concentrations). In this context, the problem of knowing the optimal mixture appears so that the maximum local lignite yield is obtained while satisfying the limits of emission of pollutants. STEM-II is used to predict the atmospheric pollution considering the weather forecast.

STEM-II is a 3D grid-based model that simulate $SO_x/NO_x/RHC$ multiphase chemistry, long-range transport and dry+wet acid deposition. This application is used for calculating the distribution of pollutants in the atmosphere from specified emission sources such as cities, power plans or forest fires under particular meteorological scenarios. The prediction of the atmospheric pollutants behaviour includes the simulation of a large set of phenomena, including diffusion, chemical transformations, advection, emission and deposition processes. A detailed mathematical description of the physical and chemical mechanisms included in this model can be found in [1].

In terms of code structure, STEM-II has a modular design that permits to split the computations of each simulated process. Figure 1 shows a simplified block diagram of the application. It consists of three major functional blocks: A simulation module, an input data module for loading the environment conditions and an output data module for exporting the simulation results. The former module includes all the computational-intensive elements required by the simulation and comprises the STEM-II kernel. More specifically, this kernel consists of a transport stage, which computes the emission and transport of the pollutants; a chemistry stage, which simulates the transformation of chemical species by gas + liquid phase chemistry and a deposition stage, which describes the dry and wet deposition of pollutants. The input data module acquires the initial chemical concentrations, topological information and meteorological data such as temperature, humidity levels or precipitation rates. Finally, the output data module exports the gaseous and aqueous concentrations of each modelled specie, as well as other relevant simulation results like reaction rates or the amount of deposited elements.

These three functional blocks are within a temporal loop that specifies the duration of the simulation. In our experiments, each time step corresponds to a minute of real time where the kernel is executed. The I/O blocks are executed for a pre-defined interval of iterations, sixty in our case.

The main motivation for developing a parallel version of this application is that in the case of Endesa power plant, STEM-II requires of meteorological input data that are available less than 12h

```
DO  x = 1, n_x
      DO  y = 1, n_y
            vertcl routine
               DO  z = 1, n_z
                        ...
               END DO
            asmm routine
               DO  z = 1, n_z
                        ...
               END DO
            rxn routine
               DO  z = 1, n_z
                        ...
               END DO
      END DO
END DO
```

Figure 2. Pseudocode of vertlq routine

before the simulation. Once those data are available, a 12h interval exists for the calculation of a 2-day prediction. In this way, it is possible to determine if future emissions will be within the limits imposed by the European norm, or if it is necessary to change the coal mixture to reduce emissions. For the sequential application, this time is too short for the computational requirements of the model, being necessary its execution on a parallel system.

This work is focused on the parallel STEM-II implementation for distributed memory systems. In [2] a detailed study of the code was performed, proving that the $vertlq$ routine (member of the chemistry module) is the most costly part of the application. The organization of this routine is shown in Figure 2. Note that it presents a nested structure, with calls to other subroutines. Dependence analysis proves that the accesses throughout the dimensions x and y do not produce data dependences. This is not the case of the rest of the dimensions[1] where different kinds of dependences appear. Based on this study, an implementation of the STEM-II for distributed memory system was performed, parallelizing both x and y loops. In [2] a detailed study of the application performance for different degrees of loop parallelism was achieved. Results show that the most efficient parallel implementation is the one that executes x loop sequentially and y loop in parallel. This behavior can be explained if we take into account the Fortran style of array storage by columns. When the x loop is executed sequentially, each processor follows the same order that the data present in memory, accessing to the data by columns with a higher degree of locality.

Figure 3 shows the structure of the parallel version, coded in Fortran 77 and using the MPI standard library for implementing communications. The root process sequentially loads and initializes the application data. Then, matrices are distributed among the processors by means a $scatter$ operation, allowing to execute in parallel the $vertlq$ routine. Part of this data are subsequently gathered to proceed to the I/O output stage and another part (that does not have to be the same) is also gath-

[1]For reasons of space only z loops are shown. There are internal nested structures where different species and phases are also traversed.

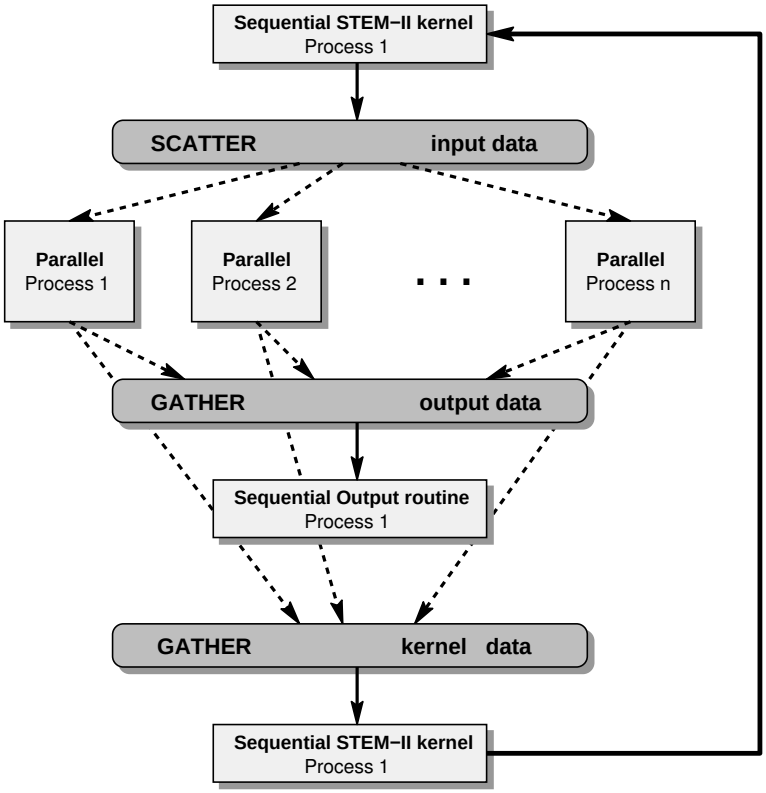

Figure 3. Diagram of the parallel implementation of STEM-II .

ered to compute the sequential version of kernel. Next section presents different alternatives for the parallelization of the I/O stage that avoid the use of gather communications.

3. STEM-II I/O parallelization

STEM-II produces a huge volume of data related to the gas-phase concentration and deposition patterns for each instant of time. In a multiprocessor environment, there are two reasons for storing part of these data in disk: the first one is the use of monitorization tools for analyzing the state of the simulation; the second reason is the use of fault-tolerance mechanisms that allow restoring a previous execution state. Before starting the simulation, the user selects the variables to be stored in disk and the frequency (in number of iterations) in which the data are exported. In both situations, the high volume of data stored to disk causes a significantly increase of the I/O cost.

Regarding the I/O input stage, there are two main reasons for not including it in our study. Firstly, the cost of this stage is small, given that the amount of loaded data are reduced. Secondly, this stage can be executed in parallel using a shared filesystem (for example Network File System) because it just includes read-only operations that do not produce access conflicts.

In this paper we propose the use of MPI I/O operations implemented in ROMIO Library [6] for performing parallel disk storage. Two techniques, called *interval* and *block* disk access, were developed and tested using PVFS distributed filesystem [7]. In contrast, their performance was compared with two local-filesystem oriented techniques called *sequential* and *local* disk access. All of them are based on the data distribution used by the parallel implementation of the STEM-II.

Now, we summarize each one of them.

- **Sequential disk access** represents the original write operation used by the parallel version of STEM-II. Figure 4(a) shows an example of this scheme. Initially, the data are distributed using a block-cyclic distribution. With a collective MPI_GATHERV communication, the root processor reconstructs the complete array. Subsequently, this processor sequentially stores the data on its private filesystem.

- **Local disk access** represents an alternative for exploiting private filesystems on a multiprocessor execution environment. In this approach (see Figure 4(b)) each processor writes its local data on its private filesystem. Given that no communication operation is required and only private filesystems are used, this operation can be performed in parallel with a high efficiency. However, it presents two main disadvantages: the original data format is not kept and the filesystems are replicated (not shared), so the global data are not accessible for the processors (including the root).

- **Interval disk access** corresponds to a parallel I/O operation on a distributed filesystem. Figure 5(a) shows the structure of this operation. Initially, each processor creates a MPI datatype related to its local distribution. This datatype includes both the entries that fit in one block of data[2] and the gap of entries that conforms the stride. Then, a MPI_FILE_SET_VIEW operation is applied to the filesystem, so that each processor only sees its assigned portion of the logical file. Finally, a MPI_FILE_WRITE operation is applied, forcing all the processors to write on different parts of the filesystem. Note that in Figure 5 we distinguish the logical view of the filesystem from the physical layout. In PVFS files are distributed using a round-robin scheme. In the figure, we assume that the stripping file size is the half of the memory block size. Interval disk access approach keeps the original data format at expense of a low locality in disk accesses. In addition, the global data are accessible for all processors.

- **Block disk access** performs a parallel I/O operation with a greater locality degree. Like interval disk access, each processor creates a datatype that represents its assigned block of data. However, the information is stored in consecutive entries of file by means of a MPI_FILE_WRITE_AT operation. Figure 5(b) illustrates an example of this technique. Note that the file layout is different from previous proposals (sequential and interval disk access) although all the stored data are still globally accessible for all processors.

In the next section the performance of these proposals is compared and the impact of effects like locality degree or filesystem architecture are analyzed.

4. Performance analysis

As test platform we used a cluster of PCs consisting of eight compute nodes interconnected by a GigaEthernet at 1000 Mb/sec. Each node consists of two Intel Pentium III at 1GHz, with 1GB of memory and 200GB of disk. The operating system is a Linux Debian 2.4.19 and the Fortran compiler is GNU f77 version 2.95.4 using optimization level -O3 and linked (if necessary) with MPICH1 (v1.2.4) libraries. We have used PVFS (v1.6.3) in our implementation of distributed filesystem with a striping factor of 64KB. Private filesystem corresponds to local $ext3$ partition of Linux.

[2]Note that the datatype can be different for each processor given that the block size can change.

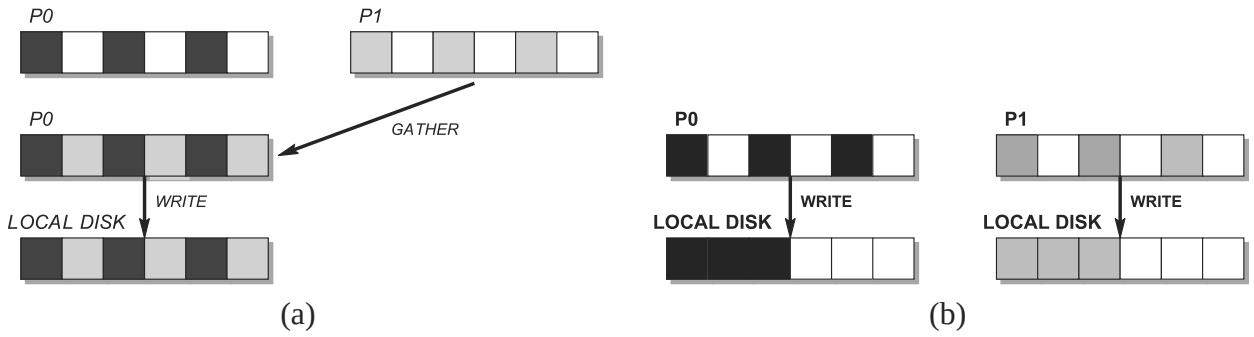

Figure 4. Examples I/O techniques on local filesystem (a) sequential disk access, (b) local disk access.

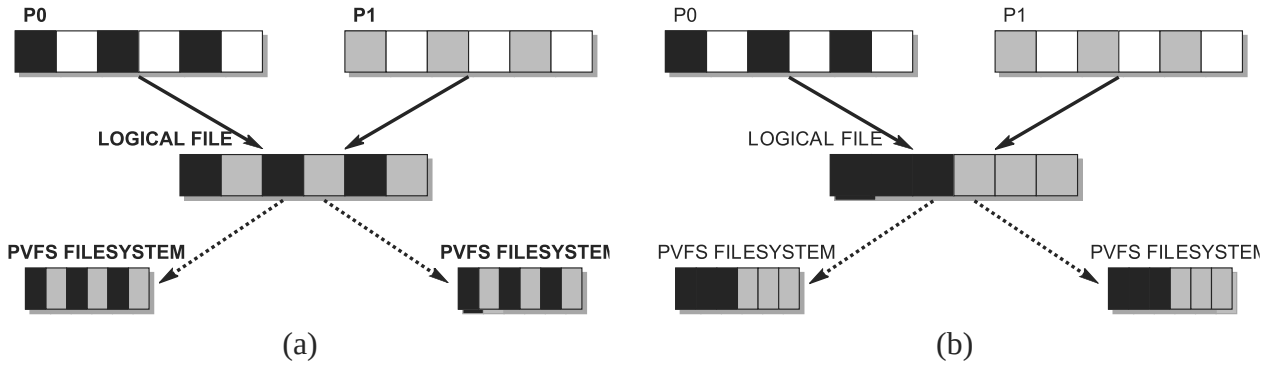

Figure 5. Examples I/O techniques on distributed filesystem (a) interval disk access, (b) block disk access.

In terms of application performance, the considered environment covers an area of 61 x 61 km^2 centred in As Pontes power plant with a resolution of 1 km^2 and 15 height levels. For this scenario different meshes are used, ranking from 61x61 (2D surface) to 61x61x15x56x3 (3D mesh extended with two dimensions related to 56 chemical species and 3 phase states). In this context, we can conclude that STEM-II is a highly expensive application in terms of amount of consumed resources, especially CPU and memory.

In this work two different sets of data were considered for being stored on disk: *Set1* consists of three 4D arrays and two 3D arrays that store the specie concentration in gas phase, summarizing 26MB of data. *Set2* adds one 5D array, up to 62MB of data. This set contains the chemical distribution of all the species considered in the simulation.

Figure 6(a) compares, for *set1*, the execution time of the initial code (called *un-optimized*) with the *interval* and *block* techniques. The un-optimized time includes the cost of the communications plus the execution time of the write operation which has a constant value of 0.19 secs. Note that for the un-optimized code the communication cost strongly increases with the number of processors. This is due to the fact that the data distribution exhibits a poor locality when a gather operation is applied. This effect is more important when the number of processors increases.

Figure 6(b) shows equivalent results for *set2*. In this case, the sequential execution time of the write operation associated to the un-optimized version has a constant value of 0.44 secs.

Several conclusions can be extracted from these graphics: Firstly, the sequential access to disk

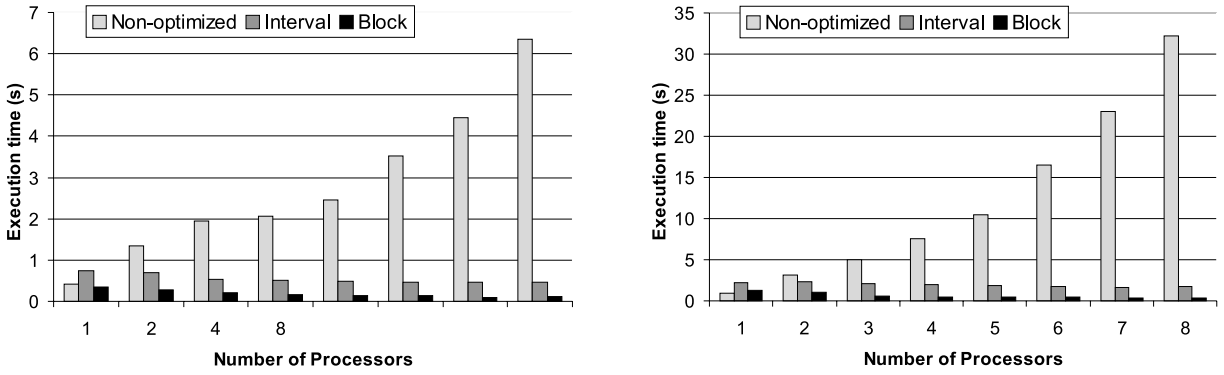

Figure 6. Execution time of un-optimized, interval and block techniques for: (a) $set1$ data, (b) $set2$ data.

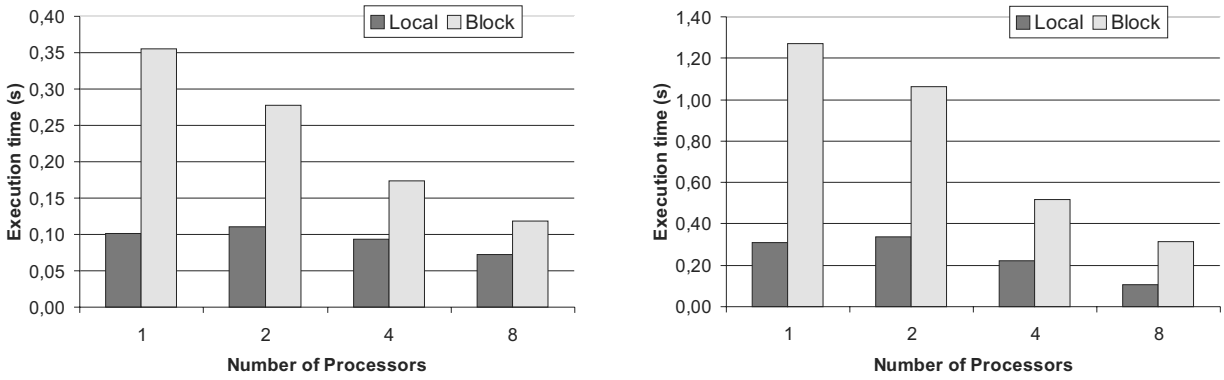

Figure 7. Execution time of local and block techniques for: (a) $set1$ data, (b) $set2$ data.

(implemented in the un-optimized version) is very inefficient due to the cost of the communications. For example, sequentially storing the contents of *set1* each iteration, represents the 30% of the overall execution time for a two processor execution. Secondly, with our proposal the I/O performance increases when the PVFS filesystem is used. Interval disk access technique implies an extra overhead respect to the sequential which causes a poor efficiency for executions with one processor. This overhead is related to the use of a distributed filesystem (instead of a local filesystem of the un-optimized version). The overhead decreases when the number of processors increases, proving the correct sustainability of the proposal.

Due to the increase of data locality, block disk access technique obtains a better scalability for all considered scenarios at spite of different file layout. Figure 7 compares the performance of this proposal with local disk access technique. We remark that the first one uses the PVFS filesystem, offering a globally accessible file whereas the last one uses a local filesystem in which only portions of the problem are stored (which makes its use impractical in most of the cases). Local disk access technique is the most efficient solution, but the performance of block technique strongly increases

as the number of processors grows. In this situation the performance tend to be the same in both proposals.

We have tested our proposals using PVFS2 (v1.1.0) distributed filesystem with both MPICH1 (v1.2.6) and MPICH2 (v1.0.2) library. Despite evaluating different optimization levels in both (MPICH and PVFS2) configurations, the performance obtained was inferior to PVFS1 filesystem, thus, the measurements were not included in this work.

5. Conclussions

In this work several parallel I/O techniques were introduced for increasing the parallelism degree of the STEM-II application. Additionally, a comparative study of local and distributed filesystem performance was presented. Results show that with the use of parallel I/O the application performance can be considerably increased. The reduction in the number of communications and the parallel disk access are the main advantages of the use of parallel I/O techniques. In addition, experimental results show that the locality degree in disk access has an important impact in the I/O performance. In case of being possible to allow transformations in the file layout, the block disk access technique is the most adequate. As future work, we propose to improve two-phase I/O techniques where the disk locality could be increased keeping small communication costs.

Acknowledgements: This work has been partially supported by the Spanish Ministry of Science and Education under the TIN2004-02156 contract.

References

[1] Carmichael, G.R., Peters, L.K., Saylor, R.D. *The STEM-II regional scale acid deposition and photochemical oxidant model - I. An overview of model development and applications.* Atmospheric Environment, 25A, 10, pp. 2077-2090, 1991.

[2] María J. Martín, David E. Singh, J. Carlos Mouriño, Francisco F. Rivera, Ramón Doallo, Javier D. Bruguera. *High performance air pollution modeling for a power plant environment.* Parallel Computing, 29, pp. 11-12, 2003.

[3] Phyllis E. Crandall, Ruth A. Aydt, Andrew A. Chien and Daniel A. Reed. *Input/Output Characteristics of Scalable Parallel Applications.* In Proceedings of Supercomputing '95, 1995.

[4] Nils Nieuwejaar, David Kotz, Apratim Purakayastha, Carla Schlatter Ellis and Michael Best. *File-Access Characteristics of Parallel Scientific Workloads.* IEEE Transactions on Parallel and Distributed Systems, 7(10), pp. 1075-1089, 1996.

[5] Fryxell B., Olson, K., Ricker P., Timmes F. X., Zingale M., Lamb D. Q., MacNeice P., Rosner R., Truran J. W. and Tufo H. *FLASH: An Adaptive Mesh Hydrodynamics Code for Modeling Astrophysical Thermonuclear Flashes.* Astrophysical Journal Supplement, 131, pp. 273-334, 2000.

[6] Rajeev Thakur, Ewing Lusk and William Gropp. *Users Guide for ROMIO: A High-Performance, Portable MPI-IO Implementation* Technical Memorandum ANL/MCS-TM-234, Mathematics and Computer Science Division, Argonne National Laboratory, 2004.

[7] P. Carns, W. Ligon III, R. Ross and R. Thakur. *PVFS: A Parallel File System for Linux Clusters*. In Proceedings of the Third Annual Linux Showcase and Conference, pp. 317-327, 2000.